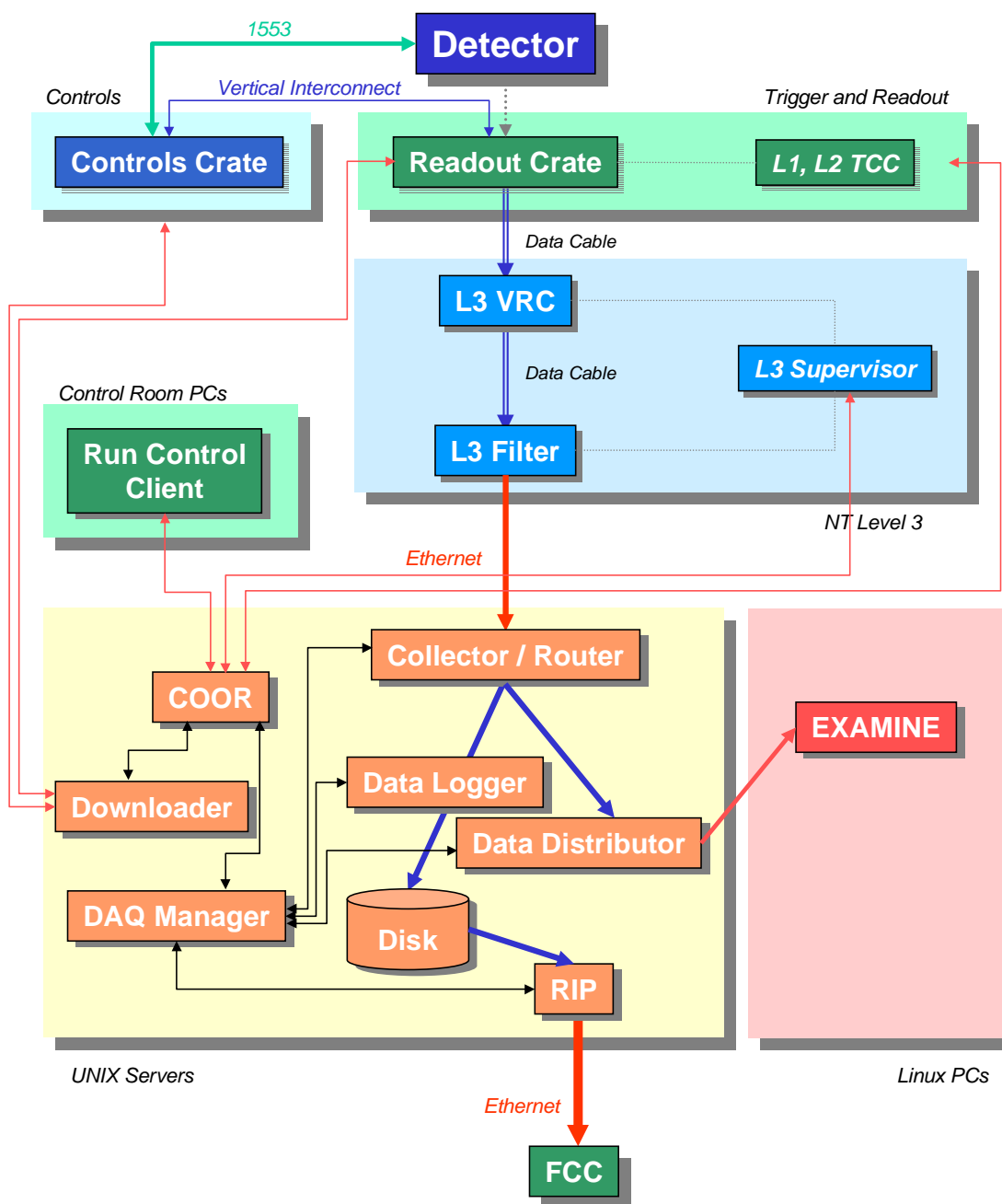


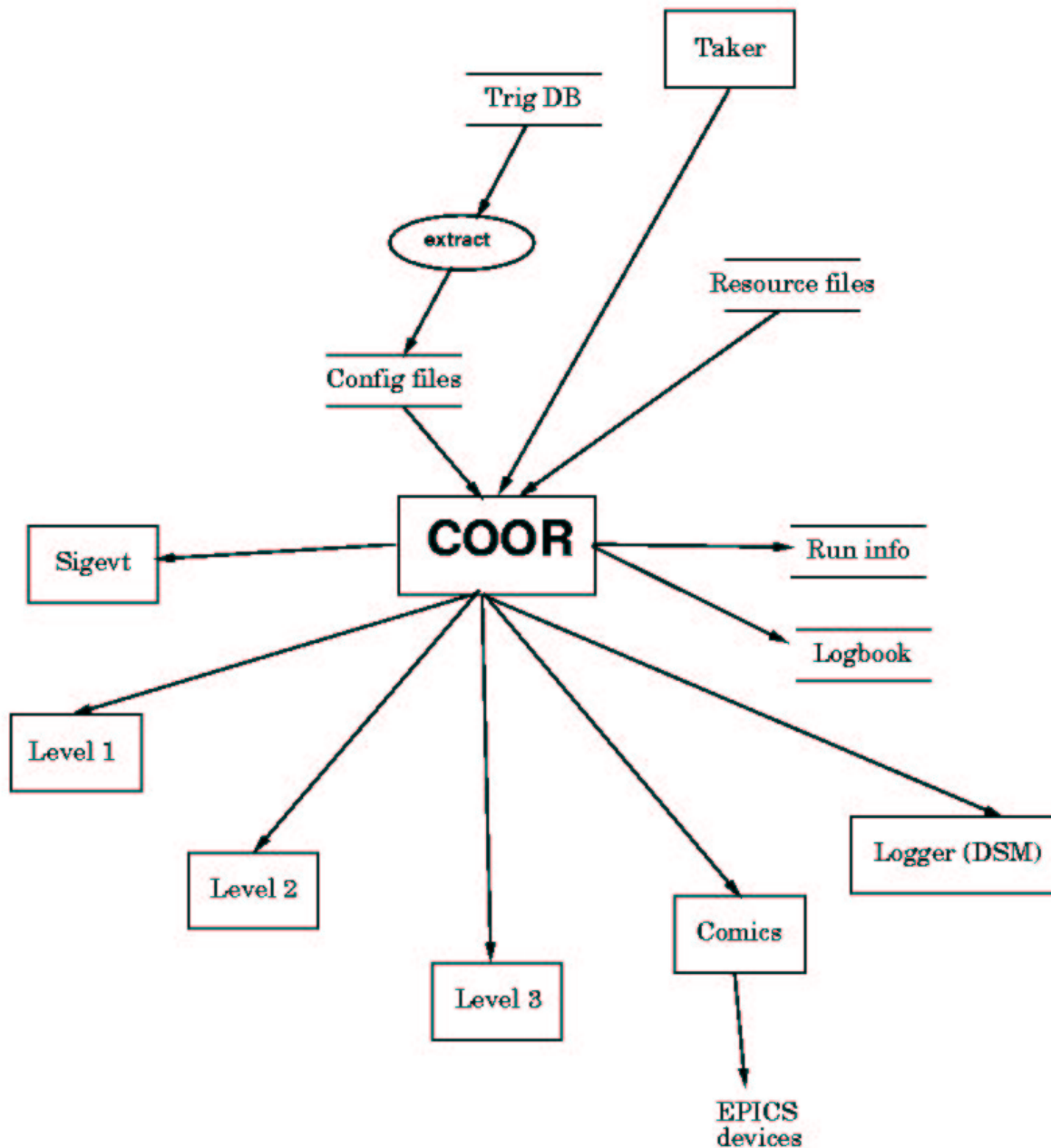
COOR

- Central “coordination” process.
- Performs run control functions:
 - Users talk to COOR to use the system.
 - Configuration and run transition requests go through COOR.
 - COOR sends commands to the other components of the system.
 - COOR maintains a model of the current detector configuration.
 - Users can allocate individual pieces of the detector for readout and control.
 - * COOR ensures that requests don’t conflict with each other.
 - COOR steps the various pieces of the system through run transitions.
 - Maintains the name database.
- Full documentation:
<http://www-d0.fnal.gov/d0dist/dist/packages/coor/devel/doc/coorover.ps>

Run Control and Configuration



COOR information flow



- Processes to which COOR sends information are called *downloaders*. COOR initiates these connections. Processes that connect to COOR to request services or information are called *clients*.

Starting and stopping

- Setup:
 - `setup d0online`
- Start:
 - `start_daq coor`
- Stop:
 - `stop_daq coor`
- Coor usually runs on `d0o1c`. It appears in `ps` listings as a process running `coormain.x`.
- Log files:
 - `/online/log/coor/*.out`
 - * Standard output/error. Stack tracebacks will appear here in the event of a crash.
 - `/online/log/coor/YYYY/MM/*.log`
 - * Daily log files. Contains detailed tracing of activities.

Control scripts

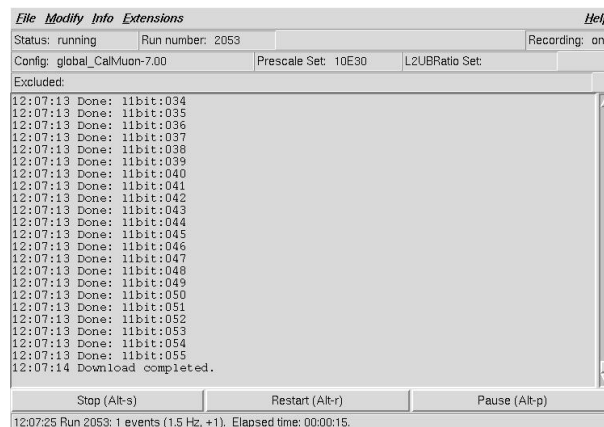
- Reinitialize:
 - `initcoor`
 - * (Will only work if no client has allocated resources.)
 - `initcoor --force`
 - * (Force all clients to give up resources they've allocated.)
 - Drops and reestablishes connections to all downloaders. Preserves all clients connections. Rereads parameter and resource files.
- Request SCL init:
 - `sclinit`
- Reinitialize L1 framework:
 - `initl1fw`
 - (No need to redo downloads.)
- Declare store beginning and end:
 - `store_begin` *store-number*
 - `store_end`

Reports

- Use 'coorinfo type' to get information about:
 - `clients` — All connected clients.
 - `crates` — All crates owned by some clients.
 - `downloaders` — Status of COOR's connections to all downloaders (and SES).
 - `itc` — All of COOR's ITC connections.
 - `l1bits` — All defined L1 trigger bits.
 - `l1egs` — All defined L1 exposure groups.
 - `l12bits` — All defined L2 trigger bits.
 - `l2tools` — All defined L2 tools/filters.
 - `l3bits` — All defined L3 trigger bits.
 - `l3clients` — All defined L3 client objects.
 - `store` — The current store.
 - `streams` — All defined streams.

Taker

- Primary user interface for controlling COOR.
- Start with 'taker'.



- Modify menu:
 - Change Trigger...
 - Free Trigger
 - Revalidate
 - Invalidate...
 - Recording
 - Change Prescales... Or L2UBRatios...
 - Prescale Set... Or L2UBRatio Set...
 - Run Parameters...
 - Reenable auto-disabled triggers
 - SCL Init
 - Exclude Crates...

Crate Exclusion

Excluded crates/devices.
Click on a button to exclude it.
(This will also exclude anything tied to it).
Click on a type name to exclude/include all devices in its group.

Sequencer	L1_Crate	<input type="checkbox"/> 0x31 fmm	<input type="checkbox"/> 0x46 ecsse	<input type="checkbox"/> 0x62 smt1_0	<input type="checkbox"/> 0x74 stt4
<input type="checkbox"/> 0x00 seq0	<input type="checkbox"/> 0x10 l1cal	<input type="checkbox"/> 0x32 fmss	<input type="checkbox"/> 0x47 ecsne	<input type="checkbox"/> 0x63 smt1_1	<input type="checkbox"/> 0x75 stt5
<input type="checkbox"/> 0x01 seq1	<input type="checkbox"/> 0x11 l1lum	<input type="checkbox"/> 0x33 fmns	<input type="checkbox"/> 0x48 ccse	<input type="checkbox"/> 0x64 smt2_0	STT_Test
<input type="checkbox"/> 0x02 seq2	<input type="checkbox"/> 0x12 l1fpd	<input type="checkbox"/> 0x34 cmwtp	<input type="checkbox"/> 0x49 ccne	<input type="checkbox"/> 0x65 smt2_1	<input type="checkbox"/> 0x76 stt6
<input type="checkbox"/> 0x03 seq3	<input type="checkbox"/> 0x13 l1ctt	<input type="checkbox"/> 0x35 cmwsp	<input type="checkbox"/> 0x4a ecnse	<input type="checkbox"/> 0x66 smt3_0	Nut_Device
<input type="checkbox"/> 0x04 seq4	<input type="checkbox"/> 0x14 l1ctm	<input type="checkbox"/> 0x36 cmwbp	<input type="checkbox"/> 0x4b ecnne	<input type="checkbox"/> 0x67 smt3_1	<input type="checkbox"/> 0x78 fch3_s
<input type="checkbox"/> 0x05 seq5	<input type="checkbox"/> 0x16 l1muc	<input type="checkbox"/> 0x37 cmwsc	Cal_TC_Crate	<input type="checkbox"/> 0x68 smt4_0	SMT_Crate
<input type="checkbox"/> 0x06 seq6	<input type="checkbox"/> 0x17 l1mun	<input type="checkbox"/> 0x38 cmetp	<input type="checkbox"/> 0x4c caltc	<input type="checkbox"/> 0x69 smt4_1	<input type="checkbox"/> 0x79 fch3_v
<input type="checkbox"/> 0x07 seq7	<input type="checkbox"/> 0x18 l1mus	<input type="checkbox"/> 0x39 cmesp	Cal_Test	<input type="checkbox"/> 0x6a smt5_0	Sequencer
DAB3_Test	<input type="checkbox"/> 0x19 l1mtm	<input type="checkbox"/> 0x3a cmebp	<input type="checkbox"/> 0x4f cal5k	<input type="checkbox"/> 0x6b smt5_1	<input type="checkbox"/> 0x7e l1caldc
<input type="checkbox"/> 0x08 test8	L2_Crate	<input type="checkbox"/> 0x3b cmesc	CFT_Crate	SIDET_Crate	Cal_PbReset
<input type="checkbox"/> 0x09 test9	<input type="checkbox"/> 0x21 l2muc	Cal_ADC_Crate	<input type="checkbox"/> 0x50 citax	<input type="checkbox"/> 0x6c sidet10	<input type="checkbox"/> pibreset
L2_Test	<input type="checkbox"/> 0x22 l2muf	<input type="checkbox"/> 0x40 ecnnw	<input type="checkbox"/> 0x51 citst	<input type="checkbox"/> 0x6d sidet01	Nut_Device
<input type="checkbox"/> 0x0a l2ta	<input type="checkbox"/> 0x23 l2cal	<input type="checkbox"/> 0x41 ecnsw	<input type="checkbox"/> 0x52 cps	STT_Crate	<input type="checkbox"/> test
<input type="checkbox"/> 0x0b l2tb	<input type="checkbox"/> 0x24 l2ps	<input type="checkbox"/> 0x42 ccnw	<input type="checkbox"/> 0x53 fps	<input type="checkbox"/> 0x70 stt0	
<input type="checkbox"/> 0x0c l2tc	<input type="checkbox"/> 0x25 l2ctt	<input type="checkbox"/> 0x43 ccsw	SMT_Crate	<input type="checkbox"/> 0x71 stt1	
<input type="checkbox"/> 0x0d l2td	Muo_Crate	<input type="checkbox"/> 0x44 ecnsw	<input type="checkbox"/> 0x60 smt0_0	<input type="checkbox"/> 0x72 stt2	
	<input type="checkbox"/> 0x30 fmsm	<input type="checkbox"/> 0x45 ecssw	<input type="checkbox"/> 0x61 smt0_1	<input type="checkbox"/> 0x73 stt3	

Muon forward north scintillator

OK (Ret) Cancel (Esc,')

- Crates may be selectively excluded from the run without having to redo the entire download.
- With no configuration loaded, can mark any crates as excluded.
- With a configuration loaded, can only exclude/unexclude crates that that configuration uses.

Coormon

- Displays current state.
- Use View menu to control what's displayed.
- Click on an item to display internal attribute values.

File Control View							
conn:alarm conn:calldnl conn:devdnl conn:dldnl conn:flldnl conn:lsdnl conn:sdagndnl							
client:d00f03.fnal.gov:1518 snyder / coormon None init (0 evts)	client:d00f05.fnal.gov:3560 d0cft / taker trackingled-cps-cftax-13-1.0 running (469 evts)	client:d00f07.fnal.gov:2100 d0run / taker calcalib-0x48-13-1.0 configured (20 evts)	client:d00f09.fnal.gov:3382 mufest / taker mufnplself-1.0 configured (4005 evts)				
client:d00f03.fnal.gov:4654 d0run / coormon None init (0 evts)	client:d00f07.fnal.gov:2083 d0run / coormon None init (0 evts)	client:d00f07.fnal.gov:4176 None / None None init (0 evts)	client:d00f0a.fnal.gov:3549 None / None None init (0 evts)				
crate:calt	crate:cmesp	crate:ecns	crate:flcal	crate:f2ctt	crate:seq0	crate:sidet10	crate:smt4_1
crate:caltc	crate:cmetp	crate:ecsne	crate:f1ctm	crate:f2glb	crate:seq1	crate:smt0_0	crate:smt5_0
crate:ccne	crate:cmwbp	crate:ecsnw	crate:f1ctt	crate:f2muc	crate:seq2	crate:smt0_1	crate:smt5_1
crate:ccnw	crate:cmwsc	crate:ecsse	crate:f1fpd	crate:f2muf	crate:seq3	crate:smt1_0	crate:stt0
crate:ccse	crate:cmwsp	crate:ecssw	crate:f1lum	crate:f2ps	crate:seq4	crate:smt1_1	crate:stt1
dev:pulser_lock dev:test							

- Control menu. (Caution: errors not reported.)
 - Flush log
 - Reconnect
 - SCL init
 - Force timeout

Common color scheme

- Connections:
 - **Green** — Connected.
 - **Red** — Not connected.
 - **Yellow** — Waiting for reply.
- Clients:
 - Neutral — Connected, but no configuration loaded.
 - **Green** — Configuration loaded.
 - **Cyan** — Running.
 - **Black** — Paused.
 - **Yellow** — Transition in progress.
 - **Red** — Cleaning up after abort or disconnect.
- Items:
 - Neutral — Not allocated.
 - **Green** — Allocated and valid.
 - **Red** — Allocated and invalid.
 - **Yellow** — Download pending.

Name Service

- COOR also maintains a database of name/value pairs.
- Serves the d0online names database (server network addresses).
- Can be used to send time-dependent information to the trigger system.
 - Example: Sending vertex information to STT.
- Can be used to selectively disable L2 inputs.
- May be freely used by detector groups.
- Names are hierarchical, separated by periods.
 - Example: `.l2_stt_beam_pos.bar0.phi`
- Names may also have properties (additional name/value pairs) associated with them.
- Property settings may be used to write name server information into the brun/erun files.
- Use `nv_editor` to browse or edit name database. (Should only be modified by experts.)

Detector model

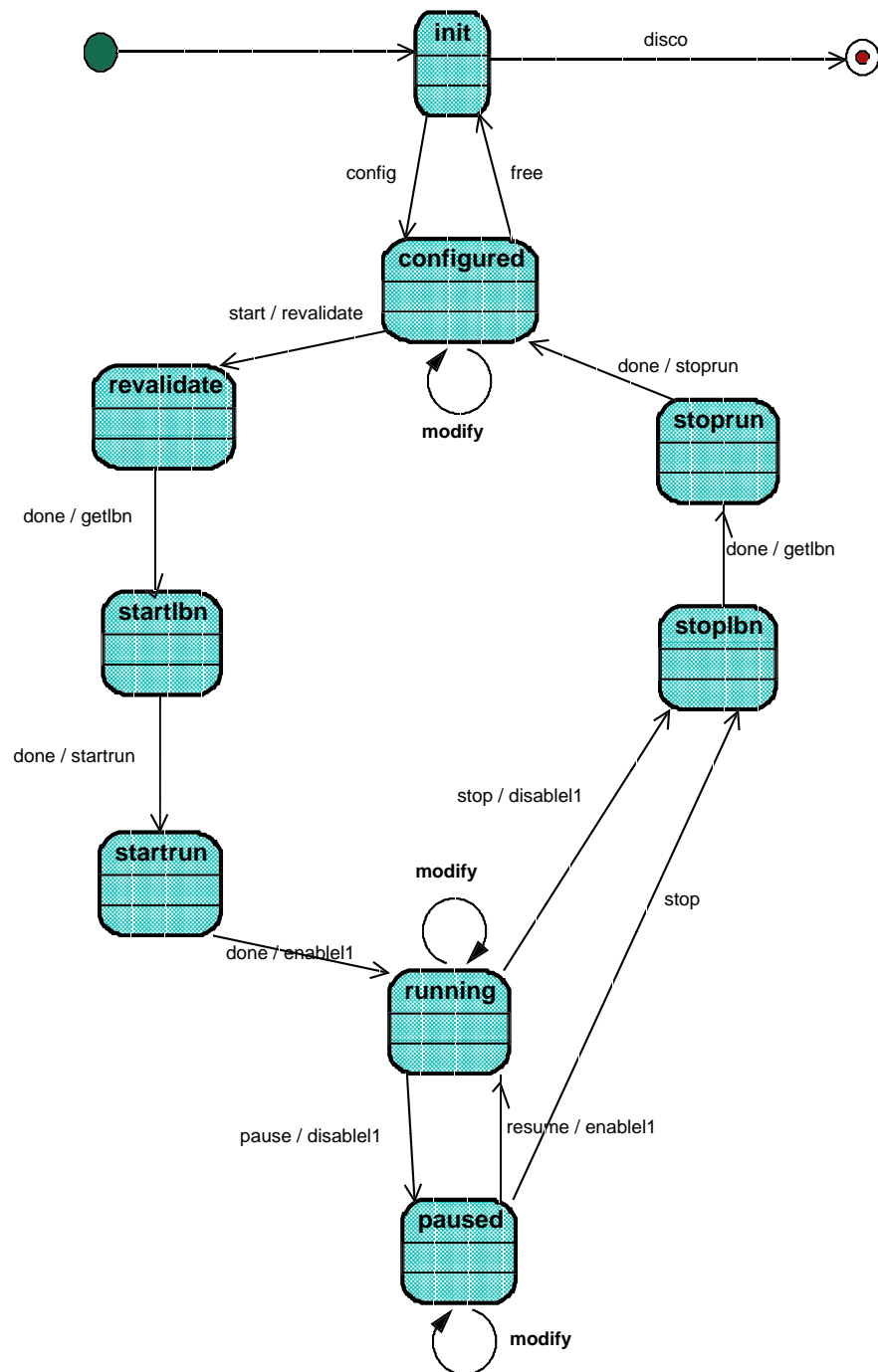
- Detector modeled by a collection of *objects*, with names of the form *class:obj*.
 - Some are permanent, some are created as needed.
- Types of objects include connections, clients, and *items* (everything else).
- Items have *attributes*. (In the common display, these start with 'd_' or 'i_'.) Object attributes are specified when a configuration is loaded.
- Items can be owned by clients. An item may be owned by multiple clients, provided they specify the same attribute values. A client may allocate an item *exclusively*, meaning no other client can allocate it. Some items are always allocated exclusively (e.g., L1 bits).
- Owned item states:
 - Valid — Item has been programmed to the requested state.
 - Invalid — Item is not known to be in the requested state; a download is required.
 - Downloading — A download for this item is pending.

Item validity

- To start a run, all items owned by a client must be valid.
- When a start run is requested, COOR attempts to make all invalid items owned by the client valid.
- To force this without starting a run, select 'Revalidate' from the taker menu.
- Use the 'Invalidate' item from the taker menu to inform coor that an item needs to be redownloaded.
- If a downloader connection breaks, the items it manages are marked as invalid.
 - Implies that, for example, when L3 is reset, you don't need to redownload the configuration — just try to start it again.

Run Transitions

- Simplified client state diagram.



Coortalk

- COOR/client communication uses text commands.
- Coortalk allows sending those commands directly.
- Start with 'coortalk'. Exit with Control-D.
- Some useful commands:
 - `help` — Get a list of commands.
 - `info type` — As earlier.
 - `disconnect dnl...` — Drop connection to *dnl*.
 - `reconnect dnl...` — Reconnect to *dnl*.
 - `force_invalidate pattern` — Invalidate items matching RE *pattern*.
 - `scl_init` — Generate a SCL init.
 - `timeout` — Force a download timeout.
 - `broadcast text` — Send a message to all clients.
 - `exitcoor` — Tell coor to exit.
 - `force_free run/client nb ...` — Force clients to give up resources.
 - `force_pause run/client nb ...` — Pause runs.
 - `force_stop run/client nb ...` — Stop runs.
 - `reinit` — Reinitialize coor, but only if there are no owned resorces.
 - `force_reinit` — Reinitialize coor.

Parameters file

- `/online/data/coor/coor.params`
- Read during startup and reinitialization.
- Taker, coormon, coortalk read it too.
- Format is Python source.
- Don't change unless requested by experts.
- Host/port addresses used by coor now come from `d0online_names.py`.
- Other file paths set here.
- Taker dialog formats defined here.

Other data files

- Resource file.
 - `/online/data/coor/resources/coor_resources.xml`
 - Read during startup and reinitialization.
 - Describes the available detector resources.
 - * Assignment of names to crates, L1 terms, etc. is done here.
 - Don't change unless requested by experts.
 - Other files in this directory define L1 trigger manager terms and L2 resources.
- Trigger configurations.
 - Live under `/online/data/coor/configurations`.
 - Canned configurations that can be selected from taker.
- Current run number.
 - `/online/data/coor/runnumber`
 - Don't change!
- Name service database.
 - `/online/data/coor/name_server_db.dat`
 - A Python pickle file.
 - Should normally never need to change this by hand.

Other Output Files

- Brun/erun/rrun files:
 - `/online/data/coor/brun`
 - COOR writes files in here on run transitions.
 - Files read by `rungrabber` and entered into the database.
- Logbook spool directory.
 - `/online/data/coor/elog_spool`
 - COOR writes files here temporarily, pending transmission to the logbook server.
 - Other processes can also write files here to be sent to the logbook. (Actual transmission may be delayed by up to 10 minutes.)

Online simulator

- Can be used to check trigger configurations.
- Run with
 - `coorsim_onl configuration`
 - *configuration* can be a file in the configuration tree, or a path to a file somewhere else.
- Will output the text that COOR would send back to Taker.
- Will create in the current directory a bunch of files containing the text that COOR would send to the downloaders, as well as logging information.

Reinits, etc.

- Many sets of instructions tell you to free the trigger list or reinit COOR when you don't really need to.
- Should only need to reinit coor the parameters file or a resource file has changed.
- (Other than general cleanliness before a store...)
- To force reinitialization of one of the processes COOR talks to, start `coortalk` and:
 - `disconnect dnl`
 - `reconnect dnl`
 - No need to free the trigger configuration — the necessary commands will be automatically resent when needed.

Other hints

- If COOR is taking a long time to respond, look on the first row of coormon. If something is yellow, that means COOR is waiting for a reply from that process — so if it stays yellow for a long time, that process may be having problems. Check its log file, etc.
- COOR will timeout after one minute if it doesn't get a response. However, there's often cleanup that has to be done afterwards that involves sending more messages — for which COOR will also wait a minute before responding. Thus, if something isn't responding, COOR can sometimes take a couple minutes to complete an operation. If you know that there is no point in waiting (because whatever COOR's waiting for will never complete successfully) you can try connecting with coortalk and issuing 'timeout'. That will force coor to time out immediately.
- If there is no apparent reason why COOR isn't responding, check the .out file in `/online/log/coor`. If there's a Python stack traceback there, COOR will need to be killed and restarted.
- Let me know if you have to kill COOR because it's misbehaving.